Forums Tutoriels Magazine FAQs Blogs Projets Chat Newsletter Études Emploi Club Contacts GO !

Developpez.com
Club des développeurs et IT pro

Accueil | ALM | Java | .NET | Dév. Web | EDI | Programmation | SGBD | Office | Solutions d'entreprise | Applications | Mobiles | Systèmes

Dév. Web | AJAX | Apache | ASP | CSS | Flash / Flex | JavaScript | PHP | Ruby | Web sémantique | Webmarketing | (X)HTML

Forum PHP | FAQ PHP | Cours PHP | Sources PHP | Livres PHP | Scripts PHP | Outils PHP | Comparatifs | Zend Framework

Gestion du XML en PHP 5 : utiliser DOM

Table des matières

Avertissements
1. Les objets de l'extension DOM
2. Le document DomDocument
2.1. Chargement
2.2. Enregistrement
2.3. Validation
3. Lire un document
3.1. L'objet DomNodeList
3.2. Rechercher et récupérer un élément
3.3. Lire les attributs
3.4. Lire les nœuds textuels
4. Modifier un document
4.1. Créer un nœud
4.2. Modifier un attribut
4.3. Insérer un nœud dans le document
4.4. Supprimer un nœud
5. Exemple simple (et inutile) : conversion XML / objets PHP
6. Fonctions avancées
7. Remerciements
Bibliographie et liens

DOM est une interface du standard XML, qui permet une gestion organisée de la structure d'un document XML. Elle est en quelque sorte complémentaire de SAX, qui permet une gestion événementielle (et qui est sans doute plus adaptée à des documents XML de grande taille). En PHP4, DOM était implémentée dans l'extension php_domxml, qui était identifiée dans la documentation comme expérimentale, et susceptible de disparaître. D'ailleurs certaines fonctionnalités, comme la validation XML/DTD, ne remplissaient pas tout à fait leur rôle... Cette extension a été remaniée pour PHP 4.3.0. En PHP5 l'extension DOM a été modifiée, et intégrée dans la distribution. Les noms de fonctions ne sont plus tout à fait les mêmes mais bien sûr l'esprit reste identique puisque c'est la même interface qui est implémentée.

Version 1.23.

L'auteur

Guillaume Piolle

L'article

Publié le 11 août 2006 -
Mis à jour le 5 novembre 2010

Version PDF Version hors-ligne

ePub, Azw et Mobi

Liens sociaux

Avertissements

Ce tutoriel suppose que vous connaissez (au moins basiquement) XML.

Les exemples de documents XML utilisés dans ce document ne doivent pas être considérés comme des modèles de structuration de données... Ils ont été choisis pour illustrer des problématiques précises relatives à l'API DOM. Nous allons travailler sur un document XML constitué d'une liste de quelques pays, classés par continents.

Les informations contenues dans ce tutoriel concernent la version 5.0.0 de PHP. Dans cette version, les fonctions DOM XML ne lancent pas d'exception DomException en cas d'erreur. De simples erreurs PHP (de niveaux divers) sont utilisées.

Ce tutoriel a vocation à évoluer, et vous êtes vivement invités à participer en contactant l'auteur pour signifier les erreurs que vous y aurez trouvées, les nouvelles sections que vous aimeriez y voir, et de manière générale toutes les modifications qui pourraient y être apportées.

1. Les objets de l'extension DOM

A la différence de l'extension de PHP4 qui était assez procédurale, l'extension DOM de PHP5 est entièrement objet. Les principales classes sont les suivantes :

- DomNode - objet nœud : documents, éléments, nœuds textuels...
- DomDocument - objet document (hérite de DomNode)
- DomElement - objet élément (hérite de DomNode)
- DomAttr - objet attribut (hérite de DomNode)
- DomNodeList - objet liste de DomNodes (ce n'est pas un tableau PHP !)

Il existe aussi des objets DomException, qui dérivent de la classe Exception de PHP5, mais la version actuelle de l'extension ne les utilise pas.

2. Le document DomDocument

Tout traitement de XML devrait commencer par la ligne suivante, qui instancie un objet DomDocument, sur lequel nous allons travailler :

Initialisation Sélectionnez

```
<?php
  $dom = new DomDocument();
?>
```

2.1. Chargement

Après l'appel du constructeur, on dispose d'un document XML vierge, sans élément racine. On peut créer les nouveaux éléments de toutes pièces, ou bien choisir de charger un document XML à partir d'un fichier sur le système de fichiers local ou à partir d'une variable chaîne de caractères. On utilise pour cela le nom du fichier XML, avec son chemin absolu ou relatif dans le système de fichiers. Chargeons le contenu de ce fichier dans notre objet *$dom* :

Chargement d'un fichier XML Sélectionnez

```
<?php
  $dom->load('fichier.xml');
?>
```

Si nous avions voulu charger le document XML à partir d'une variable (ou une chaîne statique) qui contient l'arbre XML :

Chargement depuis une chaîne XML Sélectionnez

```
<?php
  $dom->loadXML($chaineXML);
?>
```

On aurait également pu ouvrir un document HTML (grâce à la méthode DomDocument::loadHtmlFile) ou importer un document depuis SimpleXML (grâce à la fonction dom_import_simplexml, qui n'est pas une méthode de DomDocument).

2.2. Enregistrement

Il nous sera également utile d'enregistrer notre document XML sur le système de fichiers. Il suffit de procéder de la manière suivante :

Enregistrement d'un document XML Sélectionnez

```
<?php
  $dom->save('nouveauFichier.xml');
?>
```

Grâce à la méthode DomDocument::saveXML, qui renvoie le document comme une chaîne de caractères, on aurait pu récupérer le document XML dans une variable PHP.

Enregistrement dans une variable Sélectionnez

```
<?php
  $chaineXML = $dom->saveXML();
?>
```

Notez que dans ce cas, on peut spécifier en paramètre une référence sur un objet DomNode, afin que seul le sous-arbre ayant cet objet pour racine soit transmis.

2.3. Validation

L'extension DOM autorise de manière très simple la validation d'un document relativement au document DTD spécifié dans le document XML :

Validation d'un document XML Sélectionnez

```
<?php
  $dom->validate();
?>
```

On notera également les méthodes DomDocument::schemaValidate, DomDocument::schemaValidateSource, DomDocument::relaxNGValidate, DomDocument::relaxNGValidateSource, qui prennent chacun un paramètre (adresse d'un fichier pour les unes, chaîne de caractères pour les autres), et qui permettent de valider le document par rapport, respectivement, à un schéma XML sur le système de fichiers, à un schéma XML dans une chaîne de caractères, à un document relaxNG sur le système de fichiers, et à un document relaxNG dans une chaîne de caractères.

Toutes ces fonctions renvoient *true* en cas de succès, *false* en cas d'échec de la validation. En cas d'échec, des erreurs PHP de niveau *Warning* sont générées, décrivant les dérives par rapport au document de référence. Essayons par exemple de valider le document suivant par rapport à la DTD qui l'accompagne :

test.xml Sélectionnez

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE continents SYSTEM "test.dtd">
<continents>
  <europe>
    <pays3>France</pays3>
    <pays>Belgique</pays>
    <pays>Espagne</pays>
```

```
  </europe>
  <asie>
    <pays>Japon</pays>
    <pays>Inde</pays>
  </asie>
  <asie />
</continents>
```

test.dtd Sélectionnez

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT continents (europe?, asie?, amerique?, afrique?, oceanie?, antarctique?)>
  <!ELEMENT europe (pays*)>
    <!ELEMENT pays (#PCDATA)>
  <!ELEMENT asie (pays*)>
  <!ELEMENT amerique (pays*)>
  <!ELEMENT afrique (pays*)>
  <!ELEMENT oceanie (pays*)>
  <!ELEMENT antarctique (pays*)>
```

On obtiendra le résultat suivant :

Warning: file:///c:/test.xml:0: Element continents content does not follow the DTD
Expecting (europe? , asie? , amerique? , afrique? , oceanie? , antarctique?), got (europe asie asie) in **c:\siteroot\index.php** on line **6**

Warning: file:///c:/test.xml:0: Element europe content does not follow the DTD
Expecting (pays)*, got (pays3 pays pays) in **c:\siteroot\index.php** on line **6**

Warning: file:///c:/test.xml:0: No declaration for element pays3 in **c:\siteroot\index.php** on line **6**

Dans le fichier XML, le document DTD peut être inclus directement, ou spécifié par SYSTEM avec un nom relatif ou absolu sur le système de fichiers, ou encore avec une URL HTTP. Dans la version PHP4 de l'extension *php_domxml*, la référence sur le système de fichiers n'était pas acceptée, et la validation ne se faisait pas correctement (notamment, la vérification des spécifiations +, ?, * pour le nombre d'éléments n'était pas faite).

3. Lire un document

3.1. L'objet DomNodeList

Tous les résultats multiples (comprenant des nœuds) que vous retournera DOM seront sous la forme d'un objet DomNodeList. Il faut bien garder à l'esprit qu'un DomNodeList **n'est pas un tableau** : il est hors de question d'accéder à ses membres avec un index entre crochets.

La classe DomNodeList implémente l'interface Iterator de PHP, ce qui veut dire qu'elle a forcément les méthodes *current*, *next*, *key*, *valid* et *rewind*. On utilise rarement ces méthodes, mais en gros ça veut dire qu'on peut parcourir un Iterator (et donc un objet DomNodeList) dans une boucle foreach. C'est un moyen de récupérer une référence sur un objet d'un DomNodeList. Attention cependant : foreach travaillera sur une copie de l'objet DomNodeList. Si vous l'utilisez pour modifier le document, les modifications seront bien effectives mais ne seront pas visible à l'intérieur du foreach.

L'autre moyen est une méthode qui ne vient pas de l'interface Iterator, mais qui est définie par DomNodeList : *item* . Elle prend pour unique paramètre un index numérique. Ainsi le code suivant :

récupération d'une référence à partir d'un DomNodeList Sélectionnez

```
<?php
  $element = $listeElements->item(0);
?>
```

récupère dans *$element* le premier objet pointé par le DomNodeList *$listeElements*. Si on fournit un mauvais index, la méthode ne renvoie rien. Si on exploite le résultat sans prendre de précautions, on récupère une erreur du style :

Notice: Trying to get property of non-object

3.2. Rechercher et récupérer un élément

Il y a plusieurs moyens de trouver des éléments. On peut récupérer l'élément racine du document (dans ce cas-là, on récupère un objet DomElement, et pas un DomNodeList, puisqu'il n'y a de toute manière qu'un seul élément racine) :

Récupération de l'élément racine Sélectionnez

```
<?php
  $racine = $dom->documentElement;
  echo $racine->nodeName;
?>
```

On notera que les objets DomNode (et par conséquent les objets DomElement) ont une propriété *nodeName* qui renvoie... le nom du nœud. Dans le cas d'un élément, c'est le nom de la balise. Dans le sens inverse de la propriété *documentElement* des objets DomDocument, les éléments ont une propriété *ownerDocument* qui est une référence sur le document.

On peut aussi chercher un élément par la valeur de son attribut de type ID, si celui-ci est spécifié dans une DTD associée et si le document a été validé (si vous voulez simplement chercher un élément selon la valeur de son attribut *id*, il faudrait passer par un objet DomXPath. Peut-être, dans une prochaine version du tutoriel...)

Recherche d'un élément Sélectionnez

```
<?php
  $cible = $dom->getElementById("cible");
?>
```

Si vous voulez faire une recherche par le nom de la balise, vous pouvez utiliser DomDocument::getElementsByTagName() ou DomElement::getElementsByTagName(). La première version fait une recherche dans tout le document, la deuxième dans les descendants de l'élément considéré. Ces fonctions retournent un objet DomNodeList.

test.xml Sélectionnez

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<continents>
  <europe>
    <pays>France</pays>
    <pays>Belgique</pays>
    <pays>Espagne</pays>
  </europe>
  <asie>
    <pays>Japon</pays>
    <pays>Inde</pays>
  </asie>
</continents>
```

recherche d'éléments Sélectionnez

```
<?php
  $dom = new DomDocument;
  $dom->load("test.xml");
  $listePays = $dom->getElementsByTagName('pays');
  foreach($listePays as $pays)
    echo $pays->firstChild->nodeValue . "<br />";

  echo "---<br />";

  $europe = $dom->getElementsByTagName('europe')->item(0);
  $listePaysEurope = $europe->getElementsByTagName('pays');
  foreach($listePaysEurope as $pays)
    echo $pays->firstChild->nodeValue . "<br />";
?>
```

Trace du script Sélectionnez

```
France
Belgique
Espagne
Japon
Inde
---
France
Belgique
Espagne
```

On notera la propriété *nodeValue* des objets DomNode, qui dans le cas de nos objets DomElement et associée à DomNode->firstChild, permet de récupérer la valeur du nœud textuel fils.

3.3. Lire les attributs

Nous allons maintenant modifier un peu notre fichier XML (à la main), pour ajouter des attributs donnant le régime politique des pays cités (on suppose que la DTD aura également été modifiée en conséquence, si l'on veut profiter de la validation) :

Sélectionnez

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE continents SYSTEM "test.dtd">
<continents>
  <europe>
    <pays regime="republique">France</pays>
    <pays regime="monarchie constitutionnelle">Belgique</pays>
    <pays regime="monarchie constitutionnelle">Espagne</pays>
  </europe>
  <asie>
    <pays regime="empire">Japon</pays>
    <pays>Inde</pays>
  </asie>
</continents>
```

Lorsqu'on dispose de l'objet élément XML qui nous intéresse, on peut lire ses attributs grâce à DomElement::getAttribute(). C'est la manière la plus simple : on passe le nom de l'attribut à récupérer en paramètre, et on récupère sa valeur. Une bonne habitude pour éviter les erreurs est de vérifier l'existence de l'attribut avec la fonction DomElement::hasAttribute(), qui prend aussi le nom de l'attribut en paramètre, et qui renvoie un booléen qui dit si l'attribut est présent ou pas.

Sélectionnez

```
<?php
```

```
  $listePays = $dom->getElementsByTagName("pays");
  foreach($listePays as $pays)
  {
    echo $pays->nodeValue;
    if ($pays->hasAttribute("regime")) {
     echo  " - " . $pays->getAttribute("regime");
    }
    echo "<br />";
  }
>
```

Ce qui nous donne en sortie :

Trace du script Sélectionnez

```
France - republique
Belgique - monarchie constitutionnelle
Espagne - monarchie constitutionnelle
Japon - empire
Inde
```

3.4. Lire les nœuds textuels

On l'a déjà vu, on peut récupérer la valeur d'un nœud textuel avec l'attribut nodeValue. D'un point de vue très général, nodeValue donne la valeur d'un nœud, à savoir le contenu pour un nœud textuel, ou la valeur d'un attribut.

On sait qu'un nœud textuel en lui-même ("France", par exemple) est le premier descendant de l'élément qui le contient (<pays>). Cependant, curiosité de DOM, appeler nodeValue sur le parent du nœud textuel revient à l'appeler sur le nœud textuel lui-même :

Sélectionnez

```
<?php
  $pays = $dom->getElementsByTagName("pays");
  foreach($pays as $c)
  {
    echo $c->nodeValue . " " . $c->firstChild->nodeValue;
    echo "<br />";
  }
?>
```

Trace du script Sélectionnez

```
France France
Belgique Belgique
Espagne Espagne
Japon Japon
Inde Inde
```

On notera que si le fichier XML est indenté, la présence d'espaces, de tabulations et de retours à la ligne génère des nœuds textuels. Soyez donc attentifs à appliquer la fonction trim (ou un équivalent) sur vos valeurs texuelles, et/ou à vérifier si votre nœud textuel est vide ou pas.

4. Modifier un document

Voyons maintenant comment modifier les différents éléments d'un document XML déjà existant.

4.1. Créer un nœud

La méthode DomDocument::createElement permet, très simplement, de créer des éléments XML, en passant en paramètre le nom du nœud.

Création d'un élément Sélectionnez

```
<?php
  $nouveauPays = $dom->createElement("pays");
?>
```

Il faut bien noter que si $nouveauPays pointe maintenant vers un nouvel élément "pays", cet élément n'est pas encore positionné dans l'arbre XML. Il a été créé, mais pas intégré au document.

Si nous voulons ajouter un nœud textuel à cet élément (pour donner un nom de pays, si nous suivons notre exemple), il faut appeler DomDocument::createTextNode pour créer le nœud textuel. La méthode prend en paramètre le texte à insérer. Encore une fois, le nœud textuel est créé, mais pas intégré au document, ni même rattaché à notre nouvel élément.

Création d'un nœud textuel Sélectionnez

```
<php
  $nomPays = $dom->createTextNode("Royaume-Uni");
?>
```

On notera aussi la présence de la méthode DomNode::cloneNode, qui crée un nouveau nœud (de n'importe quelle type) par copie

d'un nœud existant :

Création d'un nœud par copie Sélectionnez

```
<?php
  $paysIdentique = $pays->cloneNode();
?>
```

Cette dernière méthode accepte un argument facultatif, un booléen (FALSE par défaut). S'il est à TRUE, tout les nœuds fils seront copiés également, et donc toute une partie de l'arborescence peut être dupliquée par ce biais.

4.2. Modifier un attribut

Il nous faut maintenant ajouter un attribut à notre nouveau nœud, afin de préciser le régime politique, conformément au reste du document XML qui nous sert d'exemple. Pour cela, nous allons utiliser DomElement::setAttribute, qui nous sert à la fois à créer un attribut et à en modifier la valeur. Le premier paramètre est le nom de l'attribut, et bien évidemment le second est sa valeur.

Création ou modification d'attribut Sélectionnez

```
<?php
  $nouveauPays->setAttribute("regime", "monarchie constitutionnelle");
?>
```

On peut supprimer un attribut avec DomElement::removeAttribute (avec le nom de l'attribut en paramètre).

4.3. Insérer un nœud dans le document

Nous avons vu comment créer les éléments et les nœuds textuels, mais encore faut-il les placer dans le document XML, et au bon endroit. L'insertion se fait par la méthode DomNode::appendChild, qui ajoute le nœud passé en paramètre à la liste des enfants du nœud sur lequel il est appelé. Le script suivant ajoute le nœud textuel $nomPays à notre nouveau nœud $nouveauPays, et ajoute ensuite celui-ci au nœud "europe".

Insertion de nouveaux éléments Sélectionnez

```
<?php
  $nouveauPays->appendChild($nomPays);
  $europe = $dom->getElementsByTagName("europe")->item(0);
  $europe->appendChild($nouveauPays);
?>
```

4.4. Supprimer un nœud

Finissons par le plus triste : vous ne voulez plus de votre nœud, et plutôt que de l'abandonner au bord de l'autoroute au départ des grandes vacances, vous allez l'euthanasier. Pour cela, vous utiliserez la terrible méthode DomNode::removeChild, en l'appelant sur le parent du nœud à supprimer et en passant en paramètre une référence sur le nœud à supprimer. Bien évidemment, tous les descendants du nœud supprimé seront également exterminés. Puisqu'il le faut, supprimons donc le nœud que nous avons eu tant de mal à créer !

Anéantissement de notre beau nœud tout neuf Sélectionnez

```
<?php
  $europe->removeChild($nouveauPays);
?>
```

5. Exemple simple (et inutile) : conversion XML / objets PHP

La fonction suivante prend en paramètre le nom d'un fichier XML accessible, le valide, et en extrait un objet PHP5 reprenant l'architecture du document XML. Chaque objet élément a quatre membres : son nom (chaîne de caractères), sa valeur CDATA (chaîne de caractères, vide au besoin), un tableau associatif "attributes" qui reprend les couples nom/valeur des attributs et un tableau "children" qui reprend les éléments fils. Les commentaires et la lecture de ce tutoriel devraient suffire à la compréhension du code. Notez que cette fonction est d'un intérêt limité, car on souhaite qu'elle s'adapte à toute forme de document, elle ne constitue donc qu'une piètre surcouche de la fonction DomDocument->load. Par contre, une fonction de ce type, spécialisée pour une classe particulière, peut être d'une grande utilité dans une application PHP5 utilisant des fichiers XML comme sources de données ou de paramétrage.

conversion XML vers objet Sélectionnez

```
<?php
  function fileToObject($fileName) {

    // création du nouvel objet document
    $dom = new DomDocument();

    // chargement à partir du fichier
    $dom->load($fileName);

    // validation à partir de la DTD référencée dans le document.
    // En cas d'erreur, on ne va pas plus loin
    if (!@$dom->validate()) {
      return false;
    }

    // création de l'objet résultat
    $object = new stdClass();
```

```
    // on référence l'adresse du fichier source
    $object->source = $fileName;

    // on récupère l'élément racine, on le met dans un membre
    // de l'objet nommé "root"
    $root = $dom->documentElement;
    $object->root = new stdClass();

    // appel d'une fonction récursive qui traduit l'élément XML
    // et passe la main à ses enfants, en parcourant tout l'arbre XML.
    getElement($root, $object->root);

    return $object;
  }
?>
```

Et voilà la fonction récursive qui parcourt l'arbre XML :

conversion XML vers objet - parcours d'arbre Sélectionnez

```
<?php
  function getElement($dom_element, $object_element) {

    // récupération du nom de l'élément
    $object_element->name = $dom_element->nodeName;

    // récupération de la valeur CDATA,
    // en supprimant les espaces de formatage.
    $object_element->textValue = trim($dom_element->firstChild->nodeValue);

    // Récupération des attributs
    if ($dom_element->hasAttributes()) {
      $object_element->attributes = array();
        foreach($dom_element->attributes as $attName=>$dom_attribute) {
          $object_element->attributes[$attName] = $dom_attribute->value;
        }
    }

    // Récupération des éléments fils, et parcours de l'arbre XML
    // on veut length >1 parce que le premier fils est toujours
    // le noeud texte
    if ($dom_element->childNodes->length > 1) {
      $object_element->children = array();
      foreach($dom_element->childNodes as $dom_child) {
        if ($dom_child->nodeType == XML_ELEMENT_NODE) {
          $child_object = new stdClass();
          getElement($dom_child, $child_object);
          array_push($object_element->children, $child_object);
        }
      }
    }
  }
?>
```

Voyons maintenant l'objet créé par cette fonction, si on l'applique au fichier XML sur lequel nous avons travaillé :

affichage de l'objet résultat Sélectionnez

```
<?php
  echo "<pre>";
  print_r(fileToObject("test.xml"));
  echo "</pre>";
?>
```

trace d'exécution Sélectionnez

```
stdClass Object
(
    [source] => test.xml
    [root] => stdClass Object
        (
            [name] => continents
            [textValue] =>
            [children] => Array
                (
                    [0] => stdClass Object
                        (
                            [name] => europe
                            [textValue] =>
                            [children] => Array
                                (
                                    [0] => stdClass Object
                                        (
                                            [name] => pays
                                            [textValue] => France
                                            [attributes] => Array
                                                (
```

```
                                                [regime] => republique
                                            )

                                        )

                                    [1] => stdClass Object
                                        (
                                            [name] => pays
                                            [textValue] => Belgique
                                            [attributes] => Array
                                                (
                                                    [regime] => monarchie constitutionnelle
                                                )

                                        )

                                    [2] => stdClass Object
                                        (
                                            [name] => pays
                                            [textValue] => Espagne
                                            [attributes] => Array
                                                (
                                                    [regime] => monarchie constitutionnelle
                                                )

                                        )

                                )

                            )

                        [1] => stdClass Object
                            (
                                [name] => asie
                                [textValue] =>
                                [children] => Array
                                    (
                                        [0] => stdClass Object
                                            (
                                                [name] => pays
                                                [textValue] => Japon
                                                [attributes] => Array
                                                    (
                                                        [regime] => empire
                                                    )

                                            )

                                        [1] => stdClass Object
                                            (
                                                [name] => pays
                                                [textValue] => Inde
                                            )

                                    )

                            )

                    )

            )

)
```

La seconde fonction que nous allons voir fait l'opération inverse : à partir d'un objet créé par la première fonction, elle écrit dans un fichier XML (à supposer que les droits d'écriture soient accordés au serveur, bien entendu).

Voilà maintenant la fonction qui prend en paramètre un objet tel que créé précédemment, et qui l'écrit sur le disque sous la forme d'un document XML. C'est la conversion inverse, en somme.

conversion objet vers XML Sélectionnez

```php
<?php
  function ObjectToFile($xmlObject) {

    // Création d'un nouvel objet document
    $dom = new DomDocument();

    // Création de l'élément racine
    $root = $dom->createElement($xmlObject->root->name);
    $dom->appendChild($root);

    // appel d'une fonction récursive qui construit l'élément XML
    // à partir de l'objet, en parcourant tout l'arbre de l'objet.
    setElement($dom, $xmlObject->root, $root);

    // Mise à jour du fichier source original
    $dom->save($xmlObject->source);
    echo $xmlObject->source;
```

```
  }
?>
```

Comme précédemment, cette fonction fait appel à une fonction récursive de parcours d'arbre en profondeur :

conversion objet vers XML - parcours d'arbre Sélectionnez

```
<?php
  function setElement($dom_document, $object_element, $dom_element) {

    // récupération de la valeur CDATA de l'élément
    if (isset($object_element->textValue)) {
      $cdata = $dom_document->createTextNode($object_element->textValue);
      $dom_element->appendChild($cdata);
    }

    // récupération des attributs
    if (isset($object_element->attributes)) {
      foreach($object_element->attributes as $attName=>$attValue) {
        $dom_element->setAttribute($attName, $attValue);
      }
    }

    // construction des éléments fils, et parcours de l'arbre
    if (isset($object_element->children)) {
      foreach($object_element->children as $childObject) {
        $child = $dom_document->createElement($childObject->name);
        setElement($dom_document, $childObject, $child);
        $dom_element->appendChild($child);
      }
    }
  }
?>
```

6. Fonctions avancées

Cette section s'étoffera au fur et à mesure de vos questions et suggestions.

7. Remerciements

Je remercie les personnes suivantes pour leur effort de relecture et/ou leurs remarques constructives : Yogui, JWhite, jeff_!, eexit. Merci aussi à jérôme et Kerod.

Je remercie les personnes suivantes pour avoir pointé diverses erreurs et avoir suggéré des corrections : Julien Boulen, Mickaël Auger, Tony Fouchard.

Bibliographie et liens

- Documentation officielle : PHP : Fonctions DOM
- Pour PHP4 : l'extention PECL DOM XML
- Pour assurer la compatibilité PHP4-DOMXML / PHP5-DOM, par Alexandre Alapetite : Transition du XML de PHP4 dom_xml à PHP5 dom
- D'autres moyens de traiter du XML en PHP : l'interface SAX, l'extension SimpleXML (seulement sur PHP5).
- Un article sur SimpleXML par Stéphane Eyskens.

Copyright © 2004-2010 Eusebius. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Responsable bénévole de la rubrique PHP : Didier Mouronval -

Developpez.com
Nous contacter
Participez
Informations légales

Services
Forum PHP
Blogs
Hébergement

Partenaires
PlanetHoster
Ikoula

Copyright © 2000-2014 - www.developpez.com